Kirchhoff Index As a Measure of Edge Centrality in Weighted Networks: Nearly Linear Time Algorithms

Huan Li[1] Zhongzhi Zhang[1]

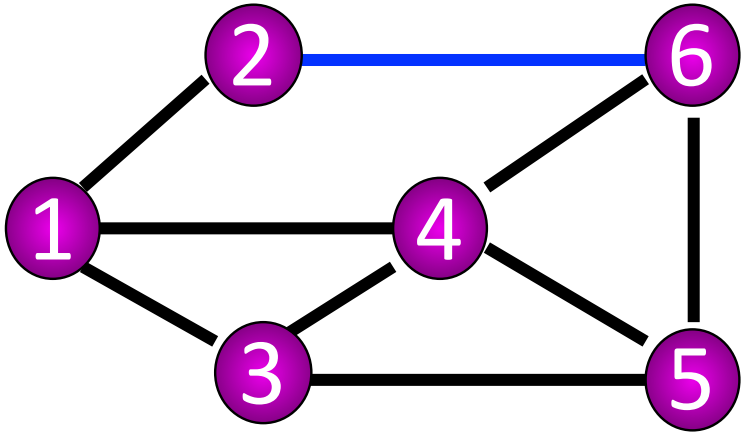
[1] School of Computer Science, Fudan University

Speaker: Huan Li

Outline

Graphs and Networks

- A graph $G = (V, E)$ with 6 vertices and 9 edges



- Modeling real-world systems by graphs

Real-world networks	Vertices	Edges
Social networks	People	Friendship/Interactions
The World Wide Web	Web pages	Web links
Power grid	Stations	Transmission lines

Centrality in Networks

Indicators of **centrality** identify the most important vertices/edges, e.g.

- Identify influential persons in a social network
  - Independent cascade model
- Identify important web pages in the World Wide Web
  - Google's PageRank algorithm
- Identify important transmission lines in a power grid

Figure: Northeast blackout of 2003 in Northern America

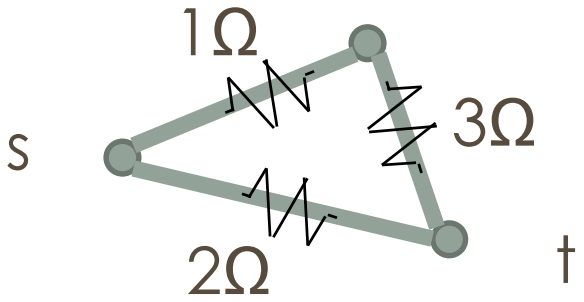
Common Used Edge Centrality Measures

- Comparing to vertex centrality, edge centrality measures are much less, due to computational challenge

Edge centrality name	Definition for an edge $e \in E$
Betweenness $\mathcal{B}(e)$	Fraction of shortest paths passing through e
Spanning edge centrality $\mathcal{S}(e)$	Fraction of spanning trees containing e
Current-flow centrality $\mathcal{F}(e)$	Average current flow passing through e

- Weakness
  - Betweenness: only considering shortest paths
  - Spanning edge centrality: lacking structure information
    - A path and a star are both trees, while their structures differ
  - Current-flow centrality: hard to compute: $\widetilde{O}(mn)$ running time

Effective Resistances (ER)



$$r(s,t) = \frac{1}{\frac{1}{2}+\frac{1}{4}} = \frac{4}{3}\,\Omega$$

- The **effective resistance** $r(s,t)$ between s,t is the *electric resistance* between them in the whole electric network
- Connection to commute times in random walks
  - Commute time $\kappa(s,t)$: expected length of random walk $s \to t \to s$
  - Relation to ER: $\kappa(s,t) = \mathrm{vol}(G) r(s,t)$
  - Thus, $r(s,t)$ takes into account all paths between s,t

Kirchhoff Index

- The **Kirchhoff index** $\mathcal{K}(G)$ is defined as

$$\mathcal{K}(G) = \sum_{u,v} r(u,v)$$

- Kirchhoff index takes into account all paths in the graph
- Kirchhoff index measures the overall connectivity of the graph
- Kirchhoff index has applications to measuring
  - The mean cost of search in a complex network
  - Robustness of first-order consensus algorithm in noisy networks
  - The global utility of social recommender systems

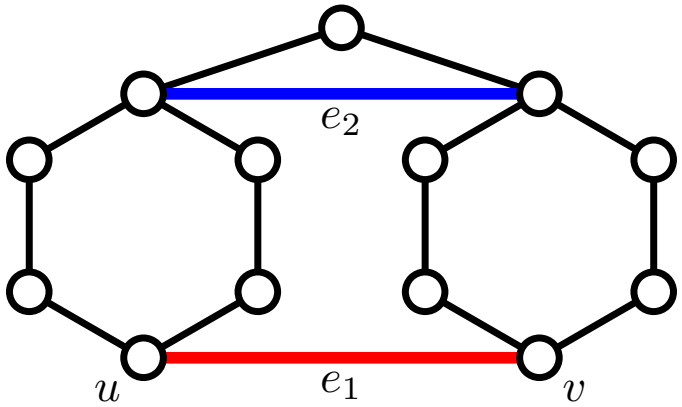
Kirchhoff Centrality

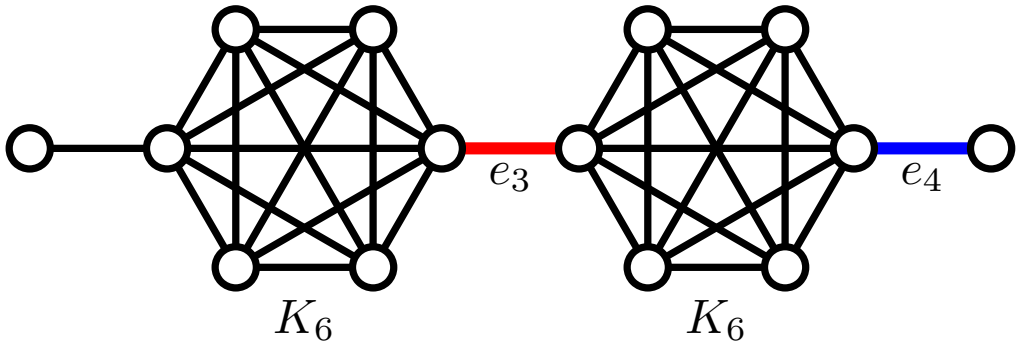
- Rayleigh's Monotonicity Law
  - $\mathcal{K}(G)$ increases when edge conductance is decreased
  - Thus, edge (partial) deletion makes the graph less connected
- An edge e's importance is reflected in the increase of Kirchhoff index when e is partially deleted
  - Resembling the power grid case
- $G\backslash_\theta e$ denotes multiplying conductance of e by $\theta \in (0, 1/2]$

θ-Kirchhoff edge centrality $\mathcal{C}_\theta(e)$	$\mathcal{C}_\theta(e) \stackrel{\text{def}}{=} \mathcal{K}(G\backslash_\theta e)$
θ-Kirchhoff edge centrality $\mathcal{C}^\Delta_\theta(e)$	$\mathcal{C}^\Delta_\theta(e) \stackrel{\text{def}}{=} \mathcal{K}(G\backslash_\theta e) - \mathcal{K}(G)$
θ-Kirchhoff vertex centrality $\mathcal{C}^\Delta_\theta(v)$	$\mathcal{C}^\Delta_\theta(v) \stackrel{\text{def}}{=} \mathcal{K}\left(G\backslash_\theta E(v)\right) - \mathcal{K}\left(G\right)$

Comparison With Other Measures

Examples





- Betweenness cannot distinguish between e_1 and e_2

$$\mathcal{B}(e_1) = \mathcal{B}(e_2) = 18$$

$$\mathcal{C}_{0.1}(e_1) = 132.65,\ \mathcal{C}_{0.1}(e_2) = 112.34$$

- Spanning edge centrality cannot distinguish between e_3 and e_4

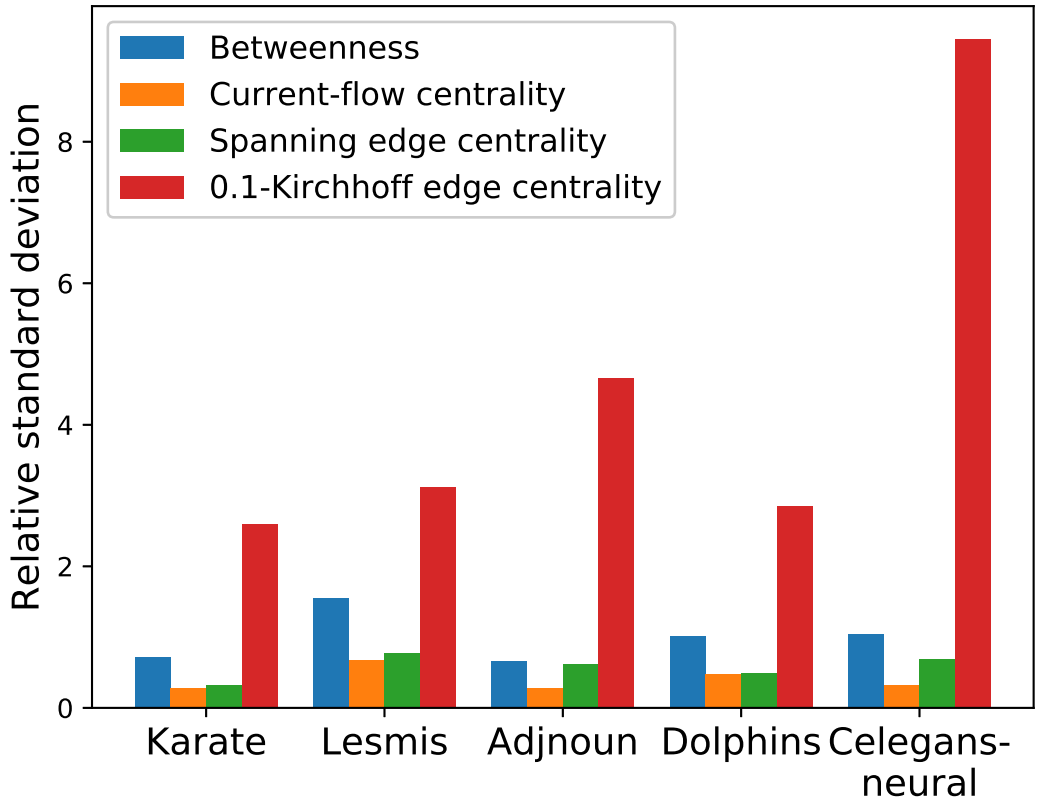
$$\mathcal{S}(e_3) = \mathcal{S}(e_4) = 1$$

$$\mathcal{C}_{0.1}(e_3) = 467.33,\ \mathcal{C}_{0.1}(e_4) = 197.33$$

Comparison With Other Measures

Experiments

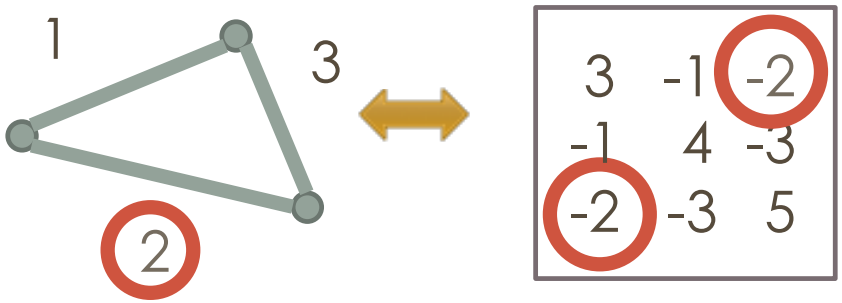
Network name	$\lvert V\rvert$	$\lvert E\rvert$
Karate	34	78
Lesmis	77	254
Adjnoun	112	425
Dolphins	62	159
Celegansneural	297	2148



$$\text{Relative standard deviation} \stackrel{\text{def}}{=} \frac{\text{Standard deviation}}{\text{Average}}$$

Outline

Graph Laplacians

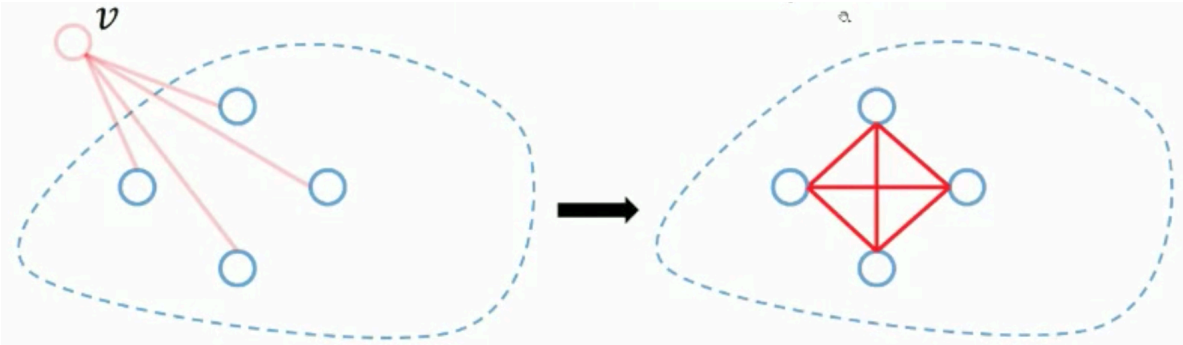


Coordinates $\Leftrightarrow$ Vertices

Nonzeros $\Leftrightarrow$ Edges

- Laplacian $\boldsymbol{L}_{[u,v]} \overset{\text{def}}{=} \begin{cases} \sum_{u \neq v} w(u,v) & u = v \\ -w(u,v) & u \neq v \end{cases}$
- $\boldsymbol{L}$ is PSD as $\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} = \sum_{(u,v) \in E} w(u,v)(\boldsymbol{x}_u - \boldsymbol{x}_v)^2$
- Defining pseudoinverse $\boldsymbol{L}^\dagger$ by inverting all nonzero eigenvalues
- $r(u,v) = (\boldsymbol{e}_u - \boldsymbol{e}_v)^T \boldsymbol{L}^\dagger (\boldsymbol{e}_u - \boldsymbol{e}_v)$, $\mathcal{K}(G) = n \operatorname{Tr}\left(\boldsymbol{L}^\dagger\right)$

Symmetric Gaussian Elimination: Additive View



$$\begin{pmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix} - \frac{1}{4} \begin{pmatrix} 4 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ 0 & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ 0 & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ 0 & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{pmatrix}$$

Gaussian Elimination for Laplacians

1. Eliminate vertex v_1 (1^{st} row and 1^{st} column) by

$$\boldsymbol{S}^{(1)} \stackrel{\text{def}}{=} \boldsymbol{L} - \frac{1}{\boldsymbol{L}_{[1,1]}} \boldsymbol{L}_{[:,1]} \boldsymbol{L}_{[:,1]}^T$$

2. $\boldsymbol{S}^{(1)}$ is called the **Schur complement** w.r.t. v_1

3. Eliminate vertices $v_1, \ldots, v_{n-1}$ by

$$\alpha_i = 1/\boldsymbol{S}_{[i,i]}^{(i-1)}, \quad \boldsymbol{c}_i = \boldsymbol{S}_{[:,i]}^{(i-1)}, \quad \boldsymbol{S}^{(i)} = \boldsymbol{S}^{(i-1)} - \alpha_i \boldsymbol{c}_i \boldsymbol{c}_i^T$$

4. **Cholesky factorization** of $\boldsymbol{L} = \boldsymbol{\mathcal{L}} \boldsymbol{\mathcal{D}} \boldsymbol{\mathcal{L}}^T$ $\left(\Rightarrow \boldsymbol{L}^\dagger = \boldsymbol{\mathcal{L}}^{-T} \boldsymbol{\mathcal{D}}^\dagger \boldsymbol{\mathcal{L}}^{-1} \right)$

$$\boldsymbol{L} = \boldsymbol{S}^{(n-1)} + \sum_{i=1}^{n-1} \alpha_i \boldsymbol{c}_i \boldsymbol{c}_i^T = \sum_{i=1}^{n-1} \alpha_i \boldsymbol{c}_i \boldsymbol{c}_i^T = \boldsymbol{\mathcal{L}} \boldsymbol{\mathcal{D}} \boldsymbol{\mathcal{L}}^T$$

where c_i is the i^{th} column of $\boldsymbol{\mathcal{L}}$, α_i is the i^{th} diagonal of $\boldsymbol{\mathcal{D}}$

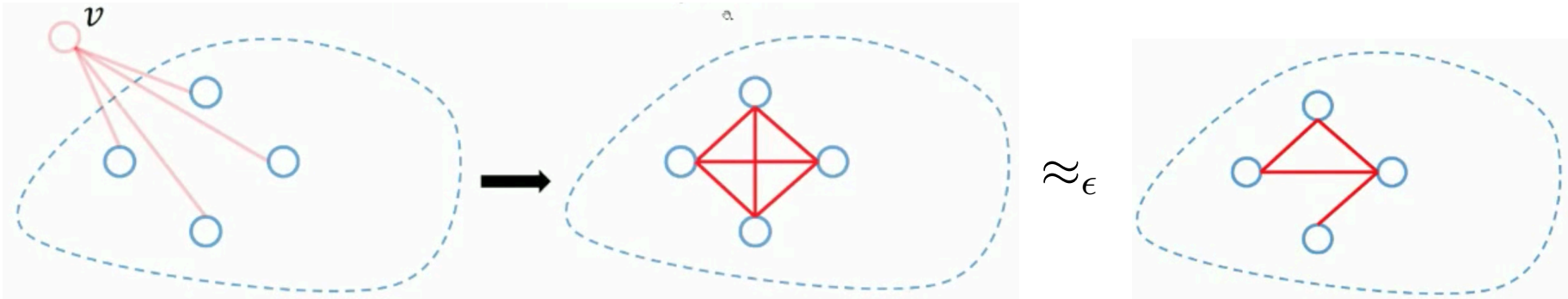
Approximating Positive Semi-Definite Matrices

- Worst-case running time $O(n^3)$
  - Approximating the dense Schur complement with a sparser graph
- $(1+\epsilon)$-approximation and ϵ-approximation ($\exp(\epsilon) \approx 1+\epsilon$ for small ϵ)

Approximation	Nonnegative scalars	PSD Matrices
$(1+\epsilon)$-approx.	$(1-\epsilon)a \leq b \leq (1+\epsilon)a$	$\forall \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \approx_{1+\epsilon} \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x}$
ϵ-approx.	$\exp(-\epsilon)a \leq b \leq \exp(\epsilon)a$	$\forall \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \approx_{\epsilon} \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x}$

- For two graphs G and H, $\boldsymbol{L}_G \approx_\epsilon \boldsymbol{L}_H$ implies
  1. All eigenvalues of $\boldsymbol{L}_G$ and $\boldsymbol{L}_H$ are similar
  2. All cut values of G and H are similar
  3. Random walks in G and H has similar mixing time and cover time
  4. The inverses are similar, i.e. $\boldsymbol{L}_G^\dagger \approx_\epsilon \boldsymbol{L}_H^\dagger$

Sparsifying Cliques in Schur Complements



- Approximate Gaussian elimination [KS16]
  - Sampling edges of the clique by upper bounds of $w(e)r(e)$ gives a $\deg(v)$-edge sparsifer
  - By eliminating vertices in a random order, computes $\boldsymbol{L} \approx_\epsilon \boldsymbol{\mathcal{L}\mathcal{D}\mathcal{L}}^T$ with $\widetilde{O}(m/\epsilon^2)$ nonzero entries in $\widetilde{O}(m/\epsilon^2)$ time
  - Enables us to ϵ-approximate $\boldsymbol{z}^T \boldsymbol{L}^\dagger \boldsymbol{z}$ in $\widetilde{O}(m/\epsilon^2)$ time

Estimating Trace of an Implicit Matrix

Hutchinson's Monte-Carlo Trace Estimation [Hut89]

1. Let $\boldsymbol{z}$ be a random ± 1 vector
2. $\mathbb{E}\left[\boldsymbol{z}^T \boldsymbol{L}^\dagger \boldsymbol{z}\right] = \mathbb{E}\left[\sum_{u,v} \boldsymbol{L}^\dagger_{[u,v]} \boldsymbol{z}_{[u]} \boldsymbol{z}_{[v]}\right] = \mathrm{Tr}\left(\boldsymbol{L}^\dagger\right)$ since
   - $\mathbb{E}\left[\boldsymbol{L}^\dagger_{[u,v]} \boldsymbol{z}_{[u]} \boldsymbol{z}_{[v]}\right] = 0$ for $u \neq v$
   - $\mathbb{E}\left[\boldsymbol{L}^\dagger_{[u,u]} \boldsymbol{z}^2_{[u]}\right] = \boldsymbol{L}^\dagger_{[u,u]}$ for $u = v$
3. Let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$ be M independent ± 1 vectors
4. $\frac{1}{M} \sum_{i=1}^{M} \boldsymbol{z}_i^T \boldsymbol{L}^\dagger \boldsymbol{z}_i$ should be close to $\mathrm{Tr}\left(\boldsymbol{L}^\dagger\right)$ when M is large
5. $\Omega(\log n/\epsilon^2)$ samples are enough to get an ϵ-approximation [AT11]
   - Derived from Johnson-Lindenstrauss lemma

Approximating $\mathcal{K}(G)$ in Nearly Linear Time

- Approximating $\mathcal{K}(G) = n\mathrm{Tr}\left(\boldsymbol{L}^{\dagger}\right)$ in $\widetilde{O}(m/\epsilon^4)$ time
  1. Compute in $\widetilde{O}(m/\epsilon^2)$ time $\boldsymbol{L} \approx_{\epsilon} \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^T$ with $\widetilde{O}(m/\epsilon^2)$ nonzeros
  2. Generate $M = O(\log n/\epsilon^2)$ independent random ± 1 vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$
  3. Return $\frac{n}{M}\sum_{i=1}^{M} \boldsymbol{z}_i^T \boldsymbol{\mathcal{L}}^{-T}\boldsymbol{\mathcal{D}}^{\dagger}\boldsymbol{\mathcal{L}}^{-1}\boldsymbol{z}_i$ as an estimate of $\mathcal{K}(G)$
- Estimating $\mathcal{C}_{\theta}(e) \stackrel{\mathrm{def}}{=} \mathcal{K}(G\backslash_{\theta} e)$ for all $e \in E$
  - Maintaining $\mathcal{K}(G)$ under $O(m)$ edge updates
  - All updates are given beforehand
  - Processing queries offline: offline div-conquer

Approximating $z^T(L\backslash_\theta e)^\dagger z$ for all $e \in E$ by Recursion

QuadEst$(L, z, E_Q = E)$: Estimate $z^T(L\backslash_\theta e)^\dagger z$ for all $e \in E_Q$ [DKPRS17]

1. Divide edges in E_Q into $E^{(1)}, E^{(2)}$ with equal sizes
2. Let C denote endpoints of edges in $E^{(1)}$ and $F = V - C$
3. Eliminate vertices in F by clique sampling in random order in nearly linear time:

$$L^\dagger \approx_\epsilon \begin{pmatrix} \mathcal{L}_{FF} & \mathbf{0} \\ \mathcal{L}_{CF} & I_{CC} \end{pmatrix}^{-T} \begin{pmatrix} \mathcal{D}^{-1} & \mathbf{0} \\ \mathbf{0} & S^\dagger \end{pmatrix} \begin{pmatrix} \mathcal{L}_{FF} & \mathbf{0} \\ \mathcal{L}_{CF} & I_{CC} \end{pmatrix}^{-1}$$

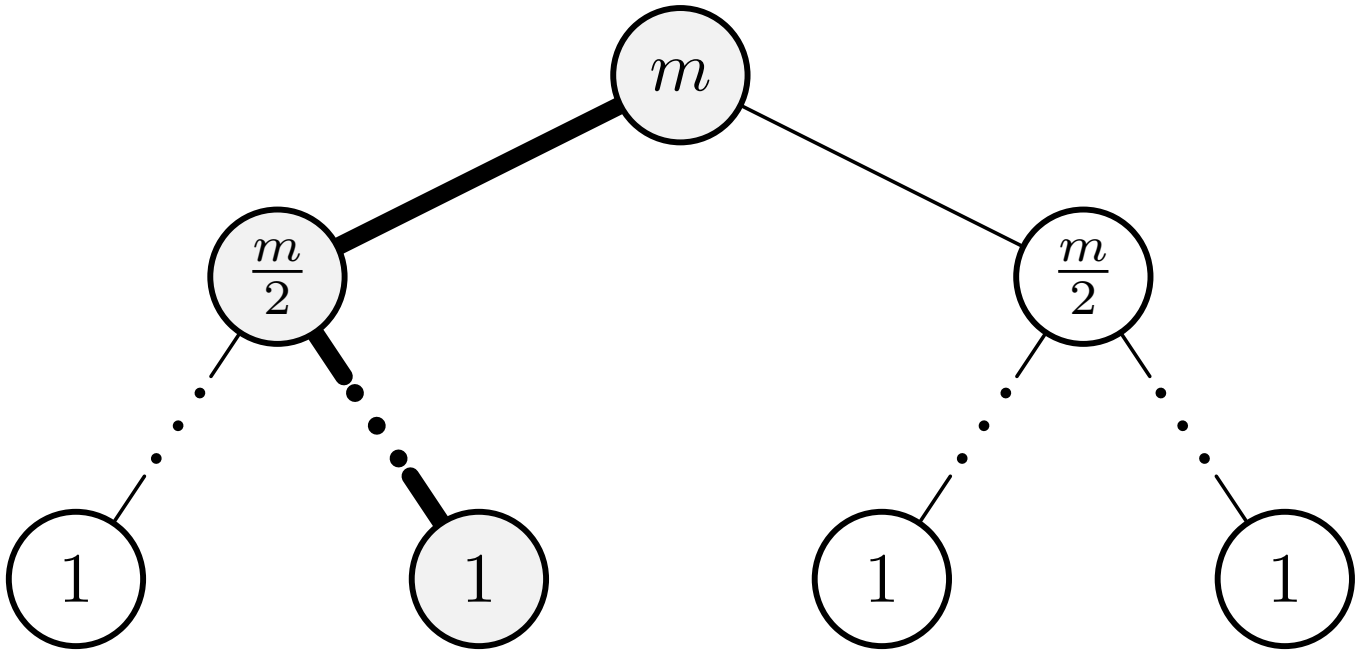
4. Let $\begin{pmatrix} \mathcal{L}_{FF} & \mathbf{0} \\ \mathcal{L}_{CF} & I_{CC} \end{pmatrix}^{-1} z \overset{\text{def}}{=} y \overset{\text{split}}{=} \begin{pmatrix} y_F \\ y_C \end{pmatrix}$, then for all $e \in E^{(1)}$

$$z^T(L\backslash_\theta e)^\dagger z = y_F^T \mathcal{D}^{-1} y_F + y_C^T (S\backslash_\theta e)^\dagger y_C$$

5. Sparsify $S\backslash E^{(1)}$ and compute $y_C^T S^\dagger y_C$ by QuadEst$(S, y_C, E^{(1)})$ recursively
6. Repeat steps 2 - 5 to $E^{(2)}$
7. Keep recursing until L only has $O(1)$ vertices, for which calculate $z^T(L\backslash_\theta e)^\dagger z$ directly for all $e \in E_Q$

Errors and Running time

Figure: Recursion tree



- Set the error of elimination to $O(\epsilon/\log n)$ everywhere
- Total running time $\widetilde{O}(m/\epsilon^4)$

Outline

Laplacian Solvers [KS16]

Solving $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$ by Iterative Refinement

1. Computes $\boldsymbol{L} \approx_{1/2} \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^T$ with $\widetilde{O}(m)$ nonzeros in $\widetilde{O}(m)$ time
2. Let $\boldsymbol{x}^{(0)} = \mathbf{0}$ and compute

$$\boldsymbol{x}^{(i+1)} \leftarrow \boldsymbol{x}^{(i)} - \frac{1}{2}\boldsymbol{\mathcal{L}}^{-T}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^{-1}(\boldsymbol{L}\boldsymbol{x}^{(i)} - \boldsymbol{b})$$

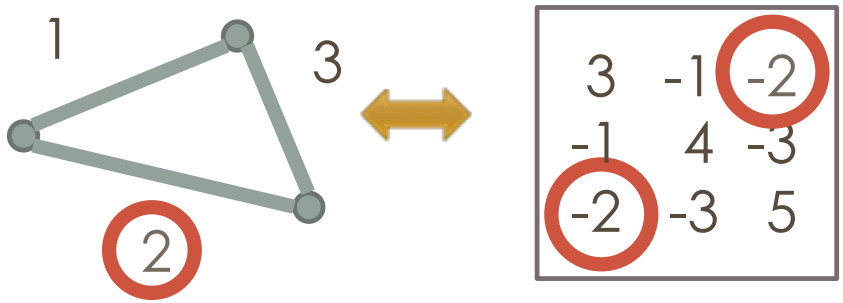
3. $\Omega(\log(1/\epsilon))$ iterations gives a solution $\hat{x}$ satisfying

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_{\boldsymbol{L}} \leq \epsilon \|\boldsymbol{x}\|_{\boldsymbol{L}}$$

   where $\|\boldsymbol{x}\|_{\boldsymbol{L}} = \sqrt{\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}}$

4. Total running time $\widetilde{O}(m \log(1/\epsilon))$

Equivalent Definitions for Laplacians



$$= \boldsymbol{D} - \boldsymbol{A}$$

$$\boldsymbol{L} = \sum_{e \in E} w(e) \boldsymbol{b}_e \boldsymbol{b}_e^T = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 2 & 0 & -2 \\ 0 & 0 & 0 \\ -2 & 0 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3 & -3 \\ 0 & -3 & 3 \end{pmatrix}$$

$$= \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B} = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix}$$

where $\boldsymbol{b}_e = \boldsymbol{b}_{u,v} = \boldsymbol{e}_u - \boldsymbol{e}_v$ for an edge $e = (u, v) \in E$, $\boldsymbol{B}^T = \begin{pmatrix} \boldsymbol{b}_1 & \cdots & \boldsymbol{b}_m \end{pmatrix}$, and $\boldsymbol{W} = \mathrm{Diag}(w(1), \ldots, w(m))$

Sherman-Morrison

- By Sherman-Morrison formula

$$\left(\boldsymbol{L}-(1-\theta)\boldsymbol{b}_e\boldsymbol{b}_e^T\right)^\dagger=\boldsymbol{L}^\dagger+(1-\theta)\frac{\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^T\boldsymbol{L}^\dagger}{1-(1-\theta)\boldsymbol{b}_e^T\boldsymbol{L}^\dagger\boldsymbol{b}_e}$$

- Approximating the numerator

$$\mathrm{Tr}\left(\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^T\boldsymbol{L}^\dagger\right)\approx_\epsilon\frac{1}{M}\sum_{i=1}^{M}\boldsymbol{z}_i^T\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^T\boldsymbol{L}^\dagger\boldsymbol{z}_i$$

  where $\boldsymbol{L}^\dagger\boldsymbol{z}_i$ can be computed by solving $\boldsymbol{L}\boldsymbol{x}=\boldsymbol{z}_i$

- **Lemma:** $\left\|\hat{\boldsymbol{y}}_i-\boldsymbol{L}^\dagger\boldsymbol{z}_i\right\|_{\boldsymbol{L}}\leq\frac{1}{36}\epsilon/n^7$ gives

$$\frac{1}{M}\sum_{i=1}^{M}\hat{\boldsymbol{y}}_i^T\boldsymbol{b}_e\boldsymbol{b}_e^T\hat{\boldsymbol{y}}_i\approx_\epsilon\mathrm{Tr}\left(\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^T\boldsymbol{L}^\dagger\right)$$

  (Derived from polynomial bounds on eigenvalues of $\boldsymbol{L}$)

Sherman-Morrison (Cont.)

- By Sherman-Morrison formula

$$\left(\boldsymbol{L} - (1-\theta)\boldsymbol{b}_e\boldsymbol{b}_e^T\right)^\dagger = \boldsymbol{L}^\dagger + (1-\theta)\frac{\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^T\boldsymbol{L}^\dagger}{1-(1-\theta)\boldsymbol{b}_e^T\boldsymbol{L}^\dagger\boldsymbol{b}_e}$$

- Approximating the denominator $1-(1-\theta)r(e)$
  - $\hat{r}(e) \approx_\epsilon r(e) \quad \Rightarrow \quad 1-(1-\theta)\hat{r}(e) \approx_{3\epsilon/\theta} 1-(1-\theta)r(e)$
  - $|\hat{r}(e) - r(e)| \leq 2\epsilon r(e)$
  - $r(e) \leq 1 \quad \Rightarrow 1-(1-\theta)r(e) \geq \theta r(e)$

- Approximating $r(e)$ for all $e \in E$ in $\widetilde{O}(m/\epsilon^2)$ time [SS11]

$$r(e) = \boldsymbol{b}_e^T\boldsymbol{L}^\dagger\boldsymbol{b}_e = \boldsymbol{b}_e^T\boldsymbol{L}^\dagger\boldsymbol{L}\boldsymbol{L}^\dagger\boldsymbol{b}_e = \boldsymbol{b}_e\boldsymbol{L}^\dagger\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{b}_e$$

$$= \mathrm{Tr}\left(\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^T\boldsymbol{L}^\dagger\boldsymbol{B}^T\right) \approx_\epsilon \sum_{i=1}^{M}\boldsymbol{z}_i^T\boldsymbol{B}\boldsymbol{L}^\dagger\boldsymbol{b}_e\boldsymbol{b}_e^T\boldsymbol{L}^\dagger\boldsymbol{B}^T\boldsymbol{z}_i$$

- Overall running time $\widetilde{O}\left(m/\left(\epsilon^2\theta^2\right)\right)$

Outline

Woodbury

- By Woodbury formula

$$\left(\boldsymbol{L} - (1-\theta)\boldsymbol{B}_{E(v)}^T \boldsymbol{B}_{E(v)}\right)^\dagger =$$
$$\boldsymbol{L}^\dagger + (1-\theta)\boldsymbol{L}^\dagger \boldsymbol{B}_{E(v)}^T \left(\boldsymbol{I} - (1-\theta)\boldsymbol{B}_{E(v)}\boldsymbol{L}^\dagger \boldsymbol{B}_{E(v)}^T\right)^{-1} \boldsymbol{B}_{E(v)}\boldsymbol{L}^\dagger$$

- By Hutchinson's trace estimation the goal becomes estimating

$$\boldsymbol{z}_i^T \boldsymbol{L}^\dagger \boldsymbol{B}_{E(v)}^T \left(\boldsymbol{I} - (1-\theta)\boldsymbol{B}_{E(v)}\boldsymbol{L}^\dagger \boldsymbol{B}_{E(v)}^T\right)^{-1} \boldsymbol{B}_{E(v)}\boldsymbol{L}^\dagger \boldsymbol{z}_i$$

  where $\boldsymbol{L}^\dagger \boldsymbol{z}_i$ can be computed by solving $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{z}_i$

- **Lemma**: $\frac{1}{36}\theta\epsilon/n^7$-approximate solutions give ϵ-approximate trace
- Need to approximate quadratic forms of

$$\left(\boldsymbol{I} - (1-\theta)\boldsymbol{B}_{E(v)}\boldsymbol{L}^\dagger \boldsymbol{B}_{E(v)}^T\right)^{-1}$$

Approximating Quadratic Forms

- $\boldsymbol{I} - (1-\theta)\boldsymbol{B}_{E(v)}\boldsymbol{L}^{\dagger}\boldsymbol{B}_{E(v)}^{T} \stackrel{\text{def}}{=} \boldsymbol{I} - \boldsymbol{A}$
  - $\lambda_{\max}(\boldsymbol{A}) \leq 1-\theta \quad \Rightarrow \quad \lambda_{\min}(\boldsymbol{I}-\boldsymbol{A}) \geq \theta$
- First-order richardson: solving $(\boldsymbol{I}-\boldsymbol{A})\boldsymbol{x} = \boldsymbol{z}_i$ in $O(1/\theta \log(1/\epsilon))$ iterations
  - Taylor expansion: $(\boldsymbol{I}-\boldsymbol{A})^{-1} = \boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \cdots$
  - Let $\hat{\boldsymbol{x}} = (\boldsymbol{I} + \boldsymbol{A} + \cdots + \boldsymbol{A}^k)\boldsymbol{z}_i$, $k = \Omega(1/\theta \log 1/\epsilon)$, then

$$\left\|\hat{\boldsymbol{x}} - (\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{z}_i\right\| \leq \epsilon \left\|(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{z}_i\right\|$$

  and

$$\langle \hat{\boldsymbol{x}}, \boldsymbol{z}_i \rangle \approx_{\epsilon} \boldsymbol{z}_i^T (\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{z}_i$$

  - $\hat{\boldsymbol{x}}$ can be computed by $\boldsymbol{x}^{(i+1)} = \boldsymbol{A}\boldsymbol{x}^{(i)} + \boldsymbol{z}_i$ by Laplacian solvers
- Chebyshev polynomial only requires $\Omega(\sqrt{1/\theta}\log(1/\epsilon))$ iterations
- However, applying $\boldsymbol{B}_{E(v)}\boldsymbol{L}^{\dagger}\boldsymbol{B}_{E(v)}^{T}$ for all $v \in V$ requires $\widetilde{O}(nm)$ time

Offline Div-conquer

- Observation: $\boldsymbol{B}_{E(v)}$ only has $2\deg(v)$ nonzeros with column indices in $N[v]$

$$\boldsymbol{B}_{E(v)}\boldsymbol{L}^{\dagger}\boldsymbol{B}_{E(v)}^{T} = \left(\boldsymbol{B}_{E(v)}\right)_{[:,N[v]]}\left(\boldsymbol{L}^{\dagger}\right)_{[N[v],N[v]]}\left(\boldsymbol{B}_{E(v)}^{T}\right)_{[N[v],:]}$$

- Define $\mathrm{SC}(\boldsymbol{L}, C) \stackrel{\text{def}}{=} \boldsymbol{S}^{(k)}$ where $F = V - C = \{v_1, \ldots, v_k\}$
  - **Fact:** $(\mathrm{SC}(\boldsymbol{L}, C))^{\dagger} = \left(\boldsymbol{L}^{\dagger}\right)_{[C,C]}$
  - **Lemma:** $\hat{\boldsymbol{S}} \approx_{\epsilon} \mathrm{SC}(\boldsymbol{L}, N[v]) \Rightarrow \boldsymbol{I} - (1-\theta)\boldsymbol{B}_{E(v)}(\hat{\boldsymbol{S}})^{\dagger}\boldsymbol{B}_{E(v)}^{T} \approx_{3\epsilon/\theta} \boldsymbol{I} - \boldsymbol{A}$
- Approximating $\mathrm{SC}(\boldsymbol{L}, N[v])$ for all $v \in V$ by offline div-conquer
  - If $\exists u$ with at least $\frac{1}{4}$ total degree, divide V into $\{u\}$ and $V - \{u\}$
  - Otherwise divide V into V_1, V_2 with at most $\frac{3}{4}$ total degree
  - Running time $T(m) = T(\frac{1}{4}m) + T(\frac{3}{4}m) + \widetilde{O}\left(m/\left(\epsilon^2\theta^2\right)\right)$
- Overall running time $\widetilde{O}\left(m/\left(\epsilon^4\theta^{2.5}\right)\right)$

Conclusions and Future Directions

- Conclusions
  - We propose three centrality measures based on Kirchhoff index
  - We show empirically that our centrality has a better discriminating power
  - We develop nearly linear time algorithms for proposed centrality measures
- Future directions
  - Improve the dependency on ϵ and θ in the running time
  - Finding k edges with maximum centrality
  - Theoretically modeling advantages/disadvantages of centrality measures

Conclusions and Future Directions

- Conclusions
  - We propose three centrality measures based on Kirchhoff index
  - We show empirically that our centrality has a better discriminating power
  - We develop nearly linear time algorithms for proposed centrality measures
- Future directions
  - Improve the dependency on ϵ and θ in the running time
  - Finding k edges with maximum centrality
  - Theoretically modeling advantages/disadvantages of centrality measures

Thank you!

Estimating Trace by JL

Lemma (Johnson-Lindenstrauss)

Given fixed vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n \in \mathbb{R}^d$ and $0 < \epsilon \leq 1/2$, let k be a positive integer such that

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n.$$

and $\boldsymbol{Q}$ be a $k \times d$ random ± 1 matrix. With probability at least $1 - 2/n$,

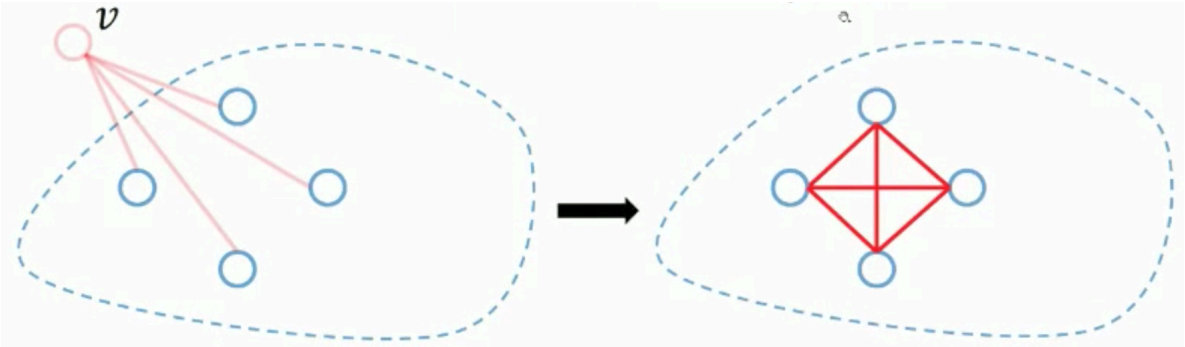
$$\forall 1 \leq i \leq n, \|\boldsymbol{v}_i\|_2^2 \approx_\epsilon \frac{1}{k} \|\boldsymbol{Q}\boldsymbol{v}_i\|_2^2 .$$

- Proof of number of samples required

$$\begin{aligned}\operatorname{Tr}(\boldsymbol{A}) = \operatorname{Tr}\left(\boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2}\right) = \left\|\boldsymbol{A}^{1/2}\right\|_F^2 \approx_\epsilon \frac{1}{k}\left\|\boldsymbol{Q}_{k\times n}\boldsymbol{A}^{1/2}\right\|_F^2 \\ = \frac{1}{k}\sum_{i=1}^{k}\left\|\boldsymbol{q}_i\boldsymbol{A}^{1/2}\right\|_2^2 = \frac{1}{k}\sum_{i=1}^{k}\left(\boldsymbol{q}_i\boldsymbol{A}^{1/2}\right)\left(\boldsymbol{q}_i\boldsymbol{A}^{1/2}\right)^T = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{q}_i\boldsymbol{A}\boldsymbol{q}_i^T\end{aligned}$$

where $\boldsymbol{q}_i$ denotes the i^{th} row of $\boldsymbol{Q}$

Clique Structure of Schur Complements



- $\boldsymbol{L}_{v_1} \overset{\text{def}}{=} \sum_{e \in N(v_1)} w(e) \boldsymbol{b}_e \boldsymbol{b}_e^T \overset{\text{split}}{=} \begin{pmatrix} d & -\boldsymbol{a}^T \\ -\boldsymbol{a} & \text{Diag}(\boldsymbol{a}) \end{pmatrix}$
- The Schur complement $\boldsymbol{S}^{(1)}$ is a Laplacian

$$\boldsymbol{S}^{(1)} \overset{\text{def}}{=} \boldsymbol{L} - \frac{1}{\boldsymbol{L}_{[1,1]}} \boldsymbol{L}_{[:,1]} \boldsymbol{L}_{[:,1]}^T = (\boldsymbol{L} - \boldsymbol{L}_{v_1}) + \left(\boldsymbol{L}_{v_1} - \frac{1}{\boldsymbol{L}_{[1,1]}} \boldsymbol{L}_{[:,1]} \boldsymbol{L}_{[:,1]}^T \right)$$

$$= \boldsymbol{L}_{G[V \setminus v_1]} + \begin{pmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & \text{Diag}(\boldsymbol{a}) - \frac{\boldsymbol{a}\boldsymbol{a}^T}{d} \end{pmatrix} = \boldsymbol{L}_{G[V \setminus v_1]} + \sum_{i \sim 1} \sum_{j \sim 1} \frac{w(1,i) w(1,j)}{d} \boldsymbol{b}_{i,j} \boldsymbol{b}_{i,j}^T$$

Clique Sampling[1]

- Graph sparsification
  - Sampling edges by $w(e)r(e)$ leads to an $\widetilde{O}(n)$-edge $\boldsymbol{H} \approx_\epsilon \boldsymbol{L}$ since

$$\sum_{e \in E} w(e)r(e) = n - 1$$

  - Sampling by $\boldsymbol{\tau}_e \geq w(e)r(e)$ leads to an $\widetilde{O}(\sum_{e \in E} \boldsymbol{\tau}_e)$-edge $\boldsymbol{H} \approx_\epsilon \boldsymbol{L}$
- Sparsifying $\boldsymbol{S}^{(1)} = \boldsymbol{L}_{G[V \setminus v_1]} + \sum_{i \sim 1} \sum_{j \sim 1} \frac{w(1,i)w(1,j)}{d} \boldsymbol{b}_{i,j} \boldsymbol{b}_{i,j}^T$
  - Schur complements preserves ER: $r^{\boldsymbol{L}}(i,j) = r^{\boldsymbol{S}^{(1)}}(i,j)$
  - $r(i,j) \leq r(1,i) + r(1,j) \leq \frac{1}{w(1,i)} + \frac{1}{w(1,j)}$
  - Sampling by $\frac{w(1,i)w(1,j)}{d} \left(\frac{1}{w(1,i)} + \frac{1}{w(1,j)} \right) = \frac{w(1,i)+w(1,j)}{d}$
  - Total edge number $\sum_{i \sim 1} \sum_{j \sim 1} \frac{w(1,i)+w(1,j)}{d} = |N(v_1)|$

[1] R. Kyng and S. Sachdeva. **FOCS'16**.

Approximate Gaussian Elimination for Laplacians[1]

- Eliminate vertices in random order
  - Every step eliminate a vertex with average degree in expectation
  - Total elimination time and total nonzero entries in factorization:

$$\frac{2m}{n} + \frac{2m}{n-1} + \ldots + \frac{2m}{2} = O(m \log n)$$

  - Thereby obtain an approximate Cholesky factorization $\boldsymbol{L} \approx_\epsilon \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^T$ with $\widetilde{O}(m/\epsilon^2)$ nonzero entries in $\widetilde{O}(m/\epsilon^2)$ time
- $\widetilde{O}(m) \stackrel{\text{def}}{=} O(m \operatorname{poly}(\log n))$ is called nearly linear time
- For any vector $\boldsymbol{z} \in \mathbb{R}^n$, we can compute in $\widetilde{O}(m/\epsilon^2)$ time an ϵ-approximation of $\boldsymbol{z}^T \boldsymbol{L}^\dagger \boldsymbol{z}$

Algorithms That Use Div-conquer

Table: Algorithms that use div-conquer

Problem	Dense $(m \approx n^2)$	Sparse $(m \approx n)$
Determinant	Exact $\widetilde{O}(n^{2.37})$[2] Approx. $\widetilde{O}(n^2)$ [3]	$\widetilde{O}(n^2)$[3]
Rand spanning tree	Exact $\widetilde{O}(n^{5/3}m^{1/3})$[4] Approx. $\widetilde{O}(n^2)$[3]	$\widetilde{O}(m^{4/3})$[5]
Approx. maxflow	$\widetilde{O}(m)$[6]	$\widetilde{O}(m)$[6]
$\boldsymbol{Lx = b}$	$\widetilde{O}(m)$[7]	$\widetilde{O}(m)$[7]

[2] V. V. Williams. **STOC'12**.

[3] D. Durfee, J. Peebles, R. Peng, and A. B. Rao. **FOCS'17**.

[4] D. Durfee, R. Kyng, J. Peebles, A. B. Rao, and S. Sachdeva. **STOC'17**.

[5] A. Madry, D. Straszak, and J. Tarnawski. **SODA'15**.

[6] R. Peng. **SODA'16**.

[7] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, and R. Peng. **STOC'14**.

Offline Div-conquer (离线分治): OI version

2008年信息学国家集训队作业　　雅礼中学　陈丹琦

从《Cash》谈一类分治算法的应用

分治算法的基本思想是将一个规模为 N 的问题分解为 K 个规模较小的子问题，这些子问题相互独立且与原问题性质相同．求出子问题的解，就可得到原问题的解．分治算法非常基础，但是分治的思想却非常重要，本文将从今年 NOI 的一道动态规划问题 Cash 开始谈如何利用分治思想来解决一类与维护决策有关的问题：

CTSC`08 Homework

(a) Dynamic programming by div-conquer

浅谈数据结构题的几个非经典解法

南京外国语学校　许昊然

CTSC`13 Report

摘 要

数据结构题是信息学竞赛中十分常见的题目，以线段树、平衡树、分块均衡、“莫队算法”为代表的经典解决方法也早耳熟能详、路人皆知。本文将介绍数据结构题的一些普及度较低，但十分有用的非经典做法。本文的主要介绍内容包括对时间分治算法及其扩展、二进制分组算法、整体二分算法，其中包含作者的很多原创内容和经验总结，希望能起到抛砖引玉的作用。

(b) Div-conquer in query timeline

SGU 529

529. It's Time to Repair the Roads

Time limit per test: 2.75 second(s)
Memory limit: 262144 kilobytes
input: standard
output: standard

Everybody knows about the problems with roads in Berland. The government has been trying to undertake major repairs for many years, but the roads have never been repaired due to the lack of money in the budget.

There are *n* cities and *m* roads in Berland. The cities are numbered from 1 to *n*. The roads are numbered from 1 to *m*. Each road connects a pair of different cities, all the roads are two-way. There is at most one road between any pair of cities. The cost of repairing is known for each road.

Clearly, repairing all roads in Berland is an unaffordable luxury, so the government decided to repair only such set of the roads, that it's possible to get from any city to any other city by the roads from this repaired set, and the total cost of these road works is minimal.

In the circumstances of the global economic crisis and global warming, road repair costs change every day. Berland's scientists managed to predict these changes, concluding that the cost of road works will change for only one road each day. They created a full list of expected changes for the coming *t* days — for each day they came up a road and its new repair cost.

(c) Maintaining MST under updates in $\widetilde{O}(m)$

CODEFORCES β
Sponsored by Telegram

Enter | Register

PROBLEMS　SUBMIT CODE　MY SUBMISSIONS　STATUS　STANDINGS　CUSTOM INVOCATION

Codeforces GYM 100551

A. Connect and Disconnect

time limit per test: 3.0 s
memory limit per test: 256 MB
input: connect.in
output: connect.out

Do you know anything about DFS, Depth First Search? For example, using this method, you can determine whether a graph is connected or not in $O(E)$ time. You can even count the number of connected components in the same time.

Do you know anything about DSU, Disjoint Set Union? Using this data structure, you can process queries like "Add an edge to the graph" and "Count the number of connected components in the graph" fast.

And do you know how to solve Dynamic Connectivity Problem? In this problem, you have to process three types of queries fast:

(d) Maintaining #CC under updates in $\widetilde{O}(m)$